Design and Analysis of Algorithms
University of California, Irvine
Prof. Aparna Chandramowlishwaran

November 17, 2016
EECS 215

Homework 3

Due on **Thursday, December 1, 2016 by 1pm**

Reading: Chapter 15

(50 Points) Edit Distance

The goal is to write a well-structured and well-documented program to compute edit distance in C/C++.

The edit distance $d(x,y)$ of two strings of text, $x[1..m]$ and $y[1..n]$, is defined to be the minimum possible cost of a sequence of transformation operations (defined below) that transforms string $x[1..m]$ into string $y[1..n]$. To define the effect of the transformation operations, we use an auxiliary string $z[1..s]$ that holds the intermediate results. At the beginning of the transformation sequence, $s = m$ and $z[1..s] = x[1..m]$ (i.e., we start with string $x[1..m]$). At the end of the transformation sequence, we should have $s = n$ and $z[1..s] = y[1..n]$ (i.e., our goal is to transform into string $y[..n]$). Throughout the transformation, we maintain the current length s of string z, as well as a cursor position i, i.e., an index into string z. The invariant $1 \leq i \leq s+1$ holds at all times during the transformation. (Notice that the cursor can move one space beyond the end of the string z in order to allow insertions at the end of the string.)

Each transformation operation may alter the string z, the size s, and the cursor position i. Each transformation operation also has an associated cost. The cost of a sequence of transformation operations is the sum of the costs of the individual operations on the sequence. The goal of the edit-distance problem is to find a sequence of transformation operations of minimum cost that transforms $x[1..m]$ into $y[1..n]$.

There are four transformation operations:

Operation	Cost	Effect
right	0	If $i = s+1$ then do nothing. Otherwise, set $i \leftarrow i+1$.
replace	4	If $i = s+1$ then do nothing. Otherwise, replace the character under the cursor by another character c by setting $z[i] \leftarrow c$, and then incrementing i.
delete	2	If $i = s+1$ then do nothing. Otherwise, delete the character c under the cursor by setting $z[i..s] \leftarrow z[i+1..s+1]$ and decrementing s. The cursor position i does not change.
insert	3	Insert the character c into string z by incrementing s, setting $z[i+1..s] \leftarrow z[i..s1]$, setting $z[i] \leftarrow c$, and then incrementing index i.

As an example, one way to transform the source string algorithm to the target string analysis is to use the sequence of operations shown below, where the position of the underlined character represents the cursor position i. Many other sequences of transformation operations also transform `algorithm` to `analysis` – the solution below is not unique. Some other solutions may cost more while some others may cost less.

Operation	z	Cost	Total
initial string	$\underline{\text{a}}$lgorithm	0	0
right	a$\underline{\text{l}}$gorithm	0	0
insert n	an$\underline{\text{l}}$gorithm	3	3
insert a	ana$\underline{\text{l}}$gorithm	3	6
right	anal$\underline{\text{g}}$orithm	0	6
replace by y	analy$\underline{\text{o}}$rithm	4	10
replace by s	analys$\underline{\text{r}}$ithm	4	14
replace by i	analysi$\underline{\text{i}}$thm	4	18
replace by s	analysis$\underline{\text{t}}$hm	4	22
delete	analysis$\underline{\text{h}}$m	2	24
delete	analysis$\underline{\text{m}}$	2	26
delete	analysis_	2	28

1. Implement your algorithm to calculate the edit distance $d(x,y)$ between two strings x and y using dynamic programming and print out the corresponding sequence of transformation operations in the style of the table above. Run your program on the strings below and submit the results.

   x = `electrical engineering` ,

   y = `computer science`.

2. Run your program on the three input files provided on Piazza. Each input file contains the following four lines:

   - The number of characters m in the string x.
   - The string x.
   - The number of characters n in the string y.
   - The string y.

   Compute the edit distance $d(x,y)$ for each input.

3. If z is implemented using an array, then the insert and delete operations require $O(n)$ time. Design a suitable data structure that allows each of the four transformation operations to be implemented in $O(1)$ time.

Submission

- When you've written up answers to all of the above questions, turn in your submission by uploading it to eee.uci.edu dropbox. LATE HOMEWORKS WILL NOT BE ACCEPTED.
- You may work in **teams of two** but submit only **one** assignment. Be sure to indicate your assignment partner in your submission.